Руководство разработчика к приложению «**********»

Разработчик:
****

Москва 2021

Оглавление

Теоретическое введение

Согласно монографии Смирнова А. В. «Анализ финансового состояния коммерческих банков» для оценки надежности и устойчивости банков и оценки эффективности банков, необходимо рассматривать следующие коэффициенты, которые рассчитываются по соответствующим формулам.

Расчет коэффициентов надежности и устойчивости банка

Коэффициент достаточности капитала, (международный стандарт), известен как коэффициент Кука: $CA = E / Anet$, где E (equity) - капитал или собственные средства

Anet - чистые активы(активы-нетто). При расчете величины коэффициента берется значение по следующей статье: форма №135, норматив Н1.0

Коэффициент мгновенной ликвидности (liquidity ratio): $LRm = LAm / Lc$

где: LAm - высоколиквидные активы, Lc - обязательства до востребования. При расчете величины коэффициента берется значение по следующей статье: форма №135, норматив Н2

Коэффициент текущей ликвидности (коэффициент покрытия; англ. current ratio, CR) Формула: Ктл = ОА / КО, где: Ктл – коэффициент текущей ликвидности;

ОА – оборотные активы КО – краткосрочные обязательства. При расчете величины коэффициента берется значение по следующей статье: форма №135, норматив Н3

Коэффициент долгосрочной ликвидности

Н4 = КДР/(К+ОД+0,5*О*)*100

Крд - кредитные требования с оставшимся сроком до даты погашения свыше 365 или 366 календарных дней, а также пролонгированные, если с учетом вновь установленных сроков погашения кредитных требований сроки, оставшиеся до их погашения, превышают 365 или 366 календарных дней, за вычетом сформированного резерва на возможные потери.

ОД - обязательства (пассивы) банка по кредитам и депозитам, полученным банком, за исключением суммы полученного банком субординированного кредита (займа, депозита) в части остаточной стоимости, включенной в расчет собственных средств (капитала) банка, а также по обращающимся на рынке долговым обязательствам банка с оставшимся сроком погашения свыше 365 или 366 календарных дней.

О* - величина минимального совокупного остатка средств по счетам со сроком исполнения обязательств до 365 календарных дней и счетам до востребования физических и юридических лиц (кроме кредитных организаций), не вошедшим в расчет показателя ОД. При расчете величины коэффициента берется значение по следующей статье: форма №135, норматив Н4

Расчет коэффициентов эффективности

Коэффициент рентабельности капитала

$ROE = NI * 360 * 100 / E*t$

где: NI-балансовая прибыль (net income), нарастающим итогом ; E - капитал банка, t - период наблюдения (дней) с начала года. NI = П – Р, где П – прибыль, Р – расходы При расчете величины берётся значение по следующей статье формы 102:

Прибыль - “19999”, Расходы - “29999”

Коэффициент рентабельности активов, ROA, (return on assets), международный стандарт): ROA = NI * 360 * 100 / Anet * t , где: NI-балансовая прибыль (net income), нарастающим итогом ; t - период наблюдения (дней) с начала года; Anet -чистые активы банка. Anet = E/ CA ,где E (equity) - капитал или собственные средства, CA- коэффициент Кука. При расчете величины коэффициента берется значение по следующей статье: форма No135, норматив Н1.0. При расчете величины капитала берется значение по следующей статье: форма No123, код показателя 000. NI= П – Р, где П – прибыль, Р – расходы При расчете величины берутся значения по следующим статьям формы 102: Прибыль - “19999”, Расходы - “29999”

Технические требования

1. Компьютер средней мощности с 8 Gb оперативной памяти и 2Gb
2. 32 или 64-битная ОС
3. Возможность установки интерпретатора Python 3.8.5
4. Возможность установки MySQL клиент-сервера

Версии библиотек

Данное приложение помимо дистрибутива Anaconda (основной для данного проекта): версия - 3.20.5, использует небольшой набор популярных библиотек питона. Ниже приведена таблица, которая описывает использованную версию каждой такой библиотеки (Табл. 1).

Библиотеки	Версия
sqlalchemy	1.4.16
PyQt5	5.15.4

PyQt5-Qt5	5.15.2
PyQt5-sip	12.9.0
numpy	1.20.3
pandas	1.2.4
python-dateutil	2.8.1
pytz	2021.1
six	1.16.0
pymysql	1.0.2
cffi	1.14.5
cryptography	3.4.7
pycparser	2.20

Таблица 1 (необходимые библиотеки)

Структура каталогов

Структура проекта описывается 19 директориями и 46 файлами. (рис.1)

```
├── Manuals
│   ├── Руководство пользователя к приложению.docx
│   └── Руководство разработчика к приложению.docx
├── Scripts
│   ├── __init__.py
│   ├── app.py
│   ├── config.py
│   ├── excel_reports
│   ├── queries
│   │   ├── authorization_db_query.py
│   │   ├── bank_db_query.py
│   │   ├── calculate
│   │   │   ├── __init__.py
│   │   │   ├── get_names.py
│   │   │   ├── get_ratios.py
│   │   │   └── ratios
│   │   │       ├── __init__.py
│   │   │       ├── efficiency.py
│   │   │       └── reliability_stability.py
│   │   └── data_mart
│   │       ├── bank
│   │       │   ├── form_bank.py
│   │       │   └── get_bank.py
│   │       ├── bank_list
│   │       │   ├── form_bank_list.py
│   │       │   └── get_bank_list.py
│   │       └── existance.py
│   ├── ui
│   │   ├── AuthorizationWindow.ui
│   │   ├── ConnectionWindow.ui
│   │   ├── admin_ui
│   │   │   ├── AdminChooseSettingsWindow.ui
│   │   │   ├── AdminReportWindow.ui
│   │   │   └── ChangeSettingsWindow.ui
│   │   ├── client_ui
│   │   │   ├── ChooseReportWindow.ui
│   │   │   └── ReportWindow.ui
│   │   └── common_ui
│   │       ├── BankListReportWindow.ui
│   │       └── BankReportWindow.ui
│   └── widgets
│       ├── __init__.py
│       ├── admin_widgets
│       │   ├── __init__.py
│       │   ├── admin_choose_settings.py
│       │   ├── report.py
│       │   └── settings
│       │       ├── __init__.py
│       │       ├── change_settings.py
│       │       ├── form_bank.py
│       │       └── form_bank_list.py
│       ├── authorization.py
│       ├── client_widgets
│       │   ├── __init__.py
│       │   ├── client_choose_report.py
│       │   ├── form_report
│       │   │   ├── __init__.py
│       │   │   ├── form_bank.py
│       │   │   └── form_bank_list.py
│       │   └── report.py
│       ├── common_widgets
│       │   ├── __init__.py
│       │   └── table_view.py
│       └── connection.py
├── requirements.txt
19 directories. 48 files
```

Рисунок 1 (содержание проекта)

Описание I уровня проекта

На I уровне проекта располагаются директории: Scripts, Manuals, requirements.txt.
Scripts – каталог, содержащий код приложения. Manuals – файлы с документацией проекта.
Requirements.txt – необходимые библиотеки для установки, чтобы запустить приложение.

Архитектура приложения

Приложение состоит из 2 пакетов, 2 модулей и 2 каталогов, которые находятся в папке Scripts. Каталог **excel_reports** содержит excel файлы, сформированных отчетов для клиентов. Запуск приложения и автоматическая загрузка окна для ввода названия соединения с БД осуществляется из модуля **App.py**. Модуль **config.py** содержит графические настройки и настройки путей приложения, название колонок в таблице отчета. Каталог **ui**, содержит описание форм виджетов, сохраненных в формате xml. Пакет **widgets** отражает логику перехода между виджетами, содержит модули и пакеты, которые имеют прямую и обратную связь между собой. Пакет **queries** содержит модули и пакеты для осуществления взаимодействия с БД (авторизация, формирование и изъятие данных из витрины данных, находящейся в виде таблице в соединенной БД). Пакеты queries и widgets имеют прямую связь между собой. Функции из пакета widgets, вызывают функции их пакета queries, для соединения с БД, графического отображения пользователю данных рассчитанных коэффициентов, получении наименований банков, авторизации пользователей. Примечание: дальнейшее описание пакетов и модулей будет осуществляться относительно папки Scripts.

Описание пакета queries

Содержание пакета queries описывается ниже (рис.2)

```
.
├── authorization_db_query.py
├── bank_db_query.py
├── calculate
│   ├── __init__.py
│   ├── get_names.py
│   ├── get_ratios.py
│   └── ratios
│       ├── __init__.py
│       ├── efficiency.py
│       ├── reliability_stability.py
│       └── tr.txt
├── data_mart
│   ├── __init__.py
│   ├── bank
│   │   ├── __init__.py
│   │   ├── form_bank.py
│   │   └── get_bank.py
│   ├── bank_list
│   │   ├── __init__.py
│   │   ├── form_bank_list.py
│   │   └── get_bank_list.py
│   └── existance.py
```

Рисунок 2 (содержимое пакета queries)

Функциональность модулей или пакетов описывается в (таб. 2).

Модуль/пакет	**Функция**
authorization_db_query.py	Модуль для соединения с базой данной и авторизации админа и клиента.
bank_db_query.py	Модуль для соединения с базой данной, формирования и извлечения витрины данных.
calculate	Пакет для расчета коэффициентов банков и извлечения наименований банков.

data_mart	Пакет для формирования и извлечения данных из витрины данных.

Таблица 2 (главные модули и пакеты queries)

Существует связь между модулем bank_db_query.py и пакетом data_mart. В модуле bank_db_query.py вызываются функции модулей из пакета data_mart для формирования запросов к БД. Существует связь между пакетами data_mart и calculate. При формировании витрины в случае, если данные еще не находятся в витрине, происходит вызов функций для расчета коэффициентов.

Листинг скрипта модулей authorization_db_query.py и bank_db_query.py. Ниже приведён список функций и docstrings каждого модуля (таб. 3).

Модуль	**Функции с докстрингами**
authorization_db_query.py	**client_verification** Аргументы: имя: str пароль: str Цель: Проверяет нахождение имени и пароля в таблице клиентов в бд. Возвращает: Булевское значение существования клиента и правильного пароля в БД. **admin_verification** Аргументы: имя: str пароль: str Цель: Проверяет нахождение имени и пароля в таблице админов в бд. Возвращает: Булевское значение существования клиента и правильного пароля в БД.

bank_db_query.py	**bank_query** Аргументы: имя банка: str начальная дата: str конечная дата: str Цель: Подсоединяется к бд, формирует витрину бд и берет из нее данные. Возвращает: Сортированный по времени список с датами, названием банка и коэффициентами из витрины данных. **bank_names_query** Аргументы: Null Цель: Подсоединяется к БД и берет из него список наименований банков. Возвращает: Сортированный список наименований банков.

Таблица 3 (Модули authorization_db_query.py, bank_db_query.py)

Описание пакета calculate

Содержание пакета calculate описывается ниже (рис. 3)

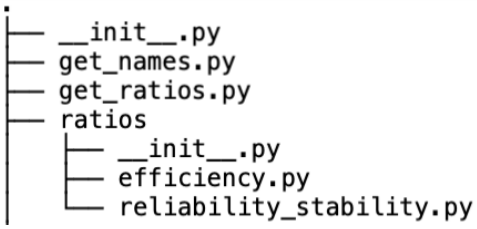

Рисунок 3 (пакет calculate)

Функциональность модулей или пакетов описывается в (таб. 4).

Модуль/пакет	**Функция**
get_names	Модуль для получения имен наименований банков из бд.
get_ratios	Расчет коэффициентов надежности и эффективности для банка или списка банков.
ratios	Пакет для расчета коэффициентов надежности или эффективности

Таблица 4 (Описание пакета calculate)

Существует связь между модулем **get_ratios** и пакетом **ratios**. Функции из модуля get_ratios вызывают функции из пакета ratios для расчетов данных. Листинг скрипта модулей get_names.py и get_ratios.py. Ниже приведён список функций и docstrings каждого модуля (таб. 5).

Модуль	**Функции с докстрингами**

get_names.py	**get_bank_names** Аргументы: соединение: sqlalchemy.engine.Connection метаданные: sqlalchemy.engine.MetaData Цель: Возвращает наименования банков из бд.
get_ratios.py	**get_bank_ratios** Аргументы: соединение: sqlalchemy.engine.Connection метаданные: sqlalchemy.engine.MetaData имя банка: str промежуток дат на начало месяца: list Цель: Рассчитывает коэффициенты надежности и эффективности для банка. Возвращает: словарь. Ключи - даты из промежутка. Значения - список рассчитанных коэффициентов за эту дату. **get_bank_list_ratios** Аргументы: соединение: sqlalchemy.engine.Connection метаданные: sqlalchemy.engine.MetaData список имен банков: list дата на начало месяца: datetime.date Цель: Рассчитывает коэффициенты надежности и эффективности для списка банков. Возвращает: словарь. Ключи - банки из списка. Значения - список рассчитанных коэффициентов за дату.

Таблица 5 (Описание модулей get_names.py и get_ratios.py)

Описание пакета ratios

Содержание пакета ratios описывается ниже (рис. 4)

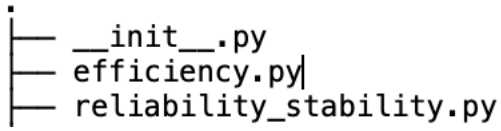

Рисунок 4 (Описание пакета ratios)

Функциональность модулей описывается в (таб. 6).

Модуль	**Функция**
efficiency.py	Расчет коэффициентов эффективности
reliability_stability.py	Расчет коэффициентов надежности и устойчивости банка.

Таблица 6 (Описание пакета ratios)

Модули efficiency.py и reliability_stability.py зависимы. Функция из модуля efficiency.py вызывает функцию из модуля reliability_stability.py для определения достаточности капитала. Листинг скрипта модулей efficiency.py и reliability_stability.py. Ниже приведён список функций и docstrings каждого модуля (таб. 7).

Модуль	**Функции с докстрингами**
reliability_stability.py	**_get_capital_adequacy** Аргументы: соединение: sqlalchemy.engine.Connection

	метаданные: sqlalchemy.engine.MetaData имя банка: str дата на начало месяца: datetime.date Цель: Рассчитывает значение коэффициента достаточности капитала на начало этой даты. Возвращает: Значение float **_get_liquidity_ratio** Аргументы: соединение: sqlalchemy.engine.Connection метаданные: sqlalchemy.engine.MetaData имя банка: str дата на начало месяца: datetime.date Цель: Рассчитывает значение коэффициента мгновенной ликвидности на начало этой даты. Возвращает: Значение float **_get_current_ratio** Аргументы: соединение: sqlalchemy.engine.Connection метаданные: sqlalchemy.engine.MetaData имя банка: str дата на начало месяца: datetime.date Цель: Рассчитывает значение коэффициента текущей ликвидности на начало этой даты. Возвращает: Значение float **_get_longterm_ratio** Аргументы: соединение: sqlalchemy.engine.Connection метаданные: sqlalchemy.engine.MetaData имя банка: str дата на начало месяца: datetime.date Цель:

	Рассчитывает значение коэффициента долгосрочной ликвидности на начало этой даты. Возвращает: Значение float **get_reliability_stability** Аргументы: соединение: sqlalchemy.engine.Connection метаданные: sqlalchemy.engine.MetaData имя банка: str дата на начало месяца: datetime.date Цель: Рассчитывает значение коэффициентов надежности и устойчивости банка на начало этой даты. Возвращает: Возвращает список этих значений.
efficiency.py	**_get_net_income_quater** Аргументы: соединение: sqlalchemy.engine.Connection метаданные: sqlalchemy.engine.MetaData имя банка: str дата на начало месяца: datetime.date Цель: Рассчитывает значение балансовой прибыли, нарастающим итогом на начало квартала (месяц начала квартала). Возвращает: Значение float на начало квартала. **_get_net_income_month** Аргументы: соединение: sqlalchemy.engine.Connection метаданные: sqlalchemy.engine.MetaData имя банка: str дата на начало месяца: datetime.date Цель: Рассчитывает значение балансовой прибыли, нарастающим итогом. Возвращает: Значение float **_get_capital** Аргументы: соединение: sqlalchemy.engine.Connection метаданные: sqlalchemy.engine.MetaData имя банка: str

	дата на начало месяца: datetime.date Цель: Рассчитывает значение капитала банка на начало этой даты Возвращает: Значение float **_get_net_assets** Аргументы: соединение: sqlalchemy.engine.Connection метаданные: sqlalchemy.engine.MetaData имя банка: str дата на начало месяца: datetime.date Цель: Рассчитывает значение чистых активов банка на начало этой даты. Возвращает: Значение float **_get_capital_profitability** Аргументы: соединение: sqlalchemy.engine.Connection метаданные: sqlalchemy.engine.MetaData имя банка: str дата на начало месяца: datetime.date Цель: Рассчитывает значение коэффициента рентабельности капитала на начало этой даты. Возвращает: Значение float **_get_return_on_assets** Аргументы: соединение: sqlalchemy.engine.Connection метаданные: sqlalchemy.engine.MetaData имя банка: str дата на начало месяца: datetime.date Цель: Рассчитывает значение коэффициент рентабельности активов на начало этой даты. Возвращает:

	Значение float **get_efficiency** Аргументы: соединение: sqlalchemy.engine.Connection метаданные: sqlalchemy.engine.MetaData имя банка: str дата на начало месяца: datetime.date Цель: Рассчитывает значение коэффициентов эффективности на начало этой даты. Возвращает: Возвращает список этих значений

Таблица 7 (Описание модулей efficiency.py и reliability_stability.py)

Описание пакета data_mart

Вернемся к описанию пакета queries (таб. 2), а именно опишем пакет data_mart. Содержание пакета описывается ниже (рис. 5)

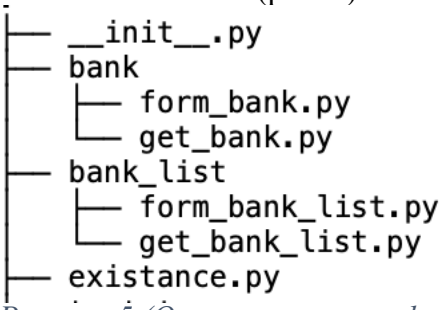

Рисунок 5 (Описание пакета data_mart)

Функциональность модулей описывается в (таб. 8).

Модуль/пакет	**Функция**
existance.py	Модуль для проверки существования витрины данных.
bank_list	Пакет для формирования и извлечения данных из витрины для списка банков
bank	Пакет для формирования и извлечения данных из витрины для списка банков

Таблица 8 (Описание пакета data_mart)

Пакеты bank_list и bank связаны с модулем existance.py. При формировании витрины проводится проверка существования витрины, в отрицательном случае, создается автоматически.

Листинг скрипта модуля existance.py. Ниже приведён список функций и docstrings каждого модуля (таб. 9).

Модуль	**Функции с докстрингами**

existance.py	**check_mart_existence** Аргументы: соединение: sqlalchemy.engine.Connection метаданные: sqlalchemy.engine.MetaData Цель: Создает витрину в БД если она не создана. Возвращает: null

Таблица 9 (Описание модуля existance.py)

Описание пакета bank

Опишем пакет bank (рис.5) Содержание пакета описывается ниже (рис. 6)

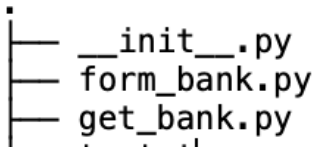

Рисунок 6 (Описание пакета bank)

Функциональность модулей описывается в (таб. 10).

Модуль	**Функция**
form_bank.py	Формирует витрину данных для банка.
get_bank.py	Извлекает данные из витрины данных для банка

Таблица 10 (Описание пакета bank)

Модули form_bank.py и get_bank.py не связаны между собой. Листинг скрипта модулей form_bank.py и get_bank.py. Ниже приведён список функций и docstrings каждого модуля (таб. 11).

Модуль	**Функции с докстрингами**
form_bank.py	**_parse_date** Аргументы: дата: str Цель: Преобразование в объект date, преобразование ее на начало следующего месяца для расчета коэффициентов. Возвращает: datetime.date **_parse_gap_dates** Аргументы: Начальная дата: str Конечная дата: str. Цель:

	Преобразование в объект date, преобразование их на начало следующего месяца для расчета коэффициентов, конвертация в список промежутка начальной и конечной даты. Возвращает: Список объектов datetime.date **_insert_bank** Аргументы: соединение: sqlalchemy.engine.Connection метаданные: sqlalchemy.engine.MetaData имя банка: str словарь: dict ключи – даты: datetime.date значения - списки, рассчитанных коэффициентов за эту дату для данного банка: list Цель: Вставляет данные в витрину данных Возвращает: Null **_mart_bank** Аргументы: соединение: sqlalchemy.engine.Connection метаданные: sqlalchemy.engine.MetaData имя банка: str начальная дата: datetime.date конечная дата: datetime.date Цель: Рассчитывает коэффициенты надежности и эффективности для банка, не включая уже рассчитанных. Возвращает: Словарь. Ключи - дата из списка дат. Значения - список рассчитанных коэффициентов для этого банка: dict Аргументы: подсистема хранения БД: sqlalchemy.engine соединение: sqlalchemy.engine.Connection метаданные: sqlalchemy.engine.MetaData имя банка: str начальная дата: str конечная дата: str Цель: Рассчитывает значения коэффициентов для банка за промежуток времени и дополняет витрину данных. Выводит рассчитанные дополнительно коэффициенты в консоль.

	Возвращает: null
get_bank.py	**_parse_date** Аргументы: дата: str Цель: Преобразование в объект date, преобразование ее на начало следующего месяца для расчета коэффициентов. Возвращает: datetime.date **_parse_gap_dates** Аргументы: Начальная дата: str Конечная дата: str. Цель: Преобразование в объект date, преобразование их на начало следующего месяца для расчета коэффициентов, конвертация в список промежутка начальной и конечной даты. Возвращает: Список объектов datetime.date **get_bank_mart** Аргументы: соединение: sqlalchemy.engine.Connection метаданные: sqlalchemy.engine.MetaData имя банка: str начальная дата: str конечная дата: str Цель: Сформировать список с датами, названием банка и коэффициентами из витрины данных. Возвращает: Сортированный по времени список

Таблица 11 (Описание модулей form_bank.py и get_bank.py)

Описание пакета bank_list

Вернемся к описанию пакета data_mart(рис. 5), а именно пакета bank_list. Содержимое пакета bank_list (рис.7)

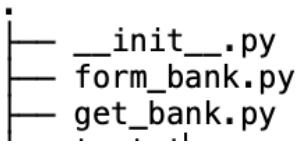

Рисунок 7 (Описание пакета bank_list)

Модули form_bank_list.py и get_bank_list.py не связаны между собой. Листинг скрипта модулей form_bank_list.py и get_bank_list.py. Ниже приведён список функций и docstrings каждого модуля (таб. 11).

Модуль	Функции с докстрингами
form_bank_list.py	**_parse_date** Аргументы: дата: str Цель: Преобразование в объект date, преобразование ее на начало следующего месяца для расчета коэффициентов. Возвращает: datetime.date **_insert_bank_list** Аргументы: соединение: sqlalchemy.engine.Connection метаданные: sqlalchemy.engine.MetaData дата коэффициентов: datatime.date словарь: dict ключи – наименования банков: str значения - списки, рассчитанных коэффициентов для банков, за данную дату: list Цель: Вставляет данные в витрину данных Возвращает: Null **_mart_bank_list** Аргументы: соединение: sqlalchemy.engine.Connection метаданные: sqlalchemy.engine.MetaData список банков: list дата: datetime.date Цель: Рассчитывает коэффициенты надежности и эффективности для списка банков, не включая уже рассчитанных. Возвращает: Словарь, ключи - даты из списка дат, значения - список рассчитанных коэффициентов для этого банка: dict **form_bank_mart** Аргументы: подсистема хранения БД: sqlalchemy.engine соединение: sqlalchemy.engine.Connection метаданные: sqlalchemy.engine.MetaData список банков: list дата: datetime.date Цель:

	Рассчитывает значения коэффициентов для списка банков за промежуток времени и дополняет витрину данных. Возвращает: null

get_bank_list.py	**_parse_date** Аргументы: дата: str Цель: Преобразование в объект date, преобразование ее на начало следующего месяца для расчета коэффициентов. Возвращает: datetime.date **get_bank_list_mart** Аргументы: соединение: sqlalchemy.engine.Connection метаданные: sqlalchemy.engine.MetaData список банков: list дата: str Цель: Сформировать список с датами, названием банка и коэффициентами из витрины данных. Возвращает: Сортированный по времени список

Таблица 12 (Описание модулей form_bank_list.py и get_bank_list.py)

Описание пакета widgets

Вернемся к описанию проекта (рис. 1), а именно пакета widgets. Переход между виджетами и логика перехода описывается в файле Manuals/Руководство к пользователю.doc. Виджеты, реализованы в виде классов и используют соответствующие по названию формы из каталога ui для графического изображения объектов.